ADT PLM

Programmer's Learning Machine

Matthieu Nicolas

IJD Seminar, 2016-02-02

Outline

Outline

Presentation of PLM

Purposes

- Application to learn programming.

Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...

Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble.

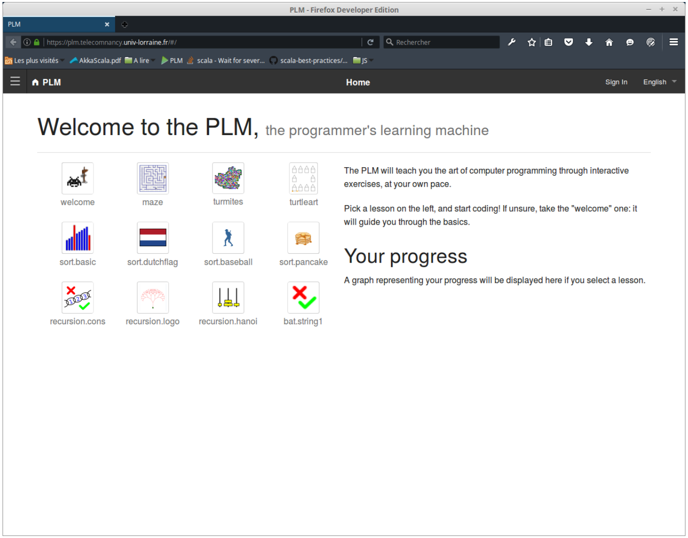
Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble.
- Used at TELECOM Nancy since 2008.

Presentation of PLM

Quick demo



Welcome to the PLM, the programmer's learning machine

The PLM will teach you the art of computer programming through interactive exercises, at your own pace.

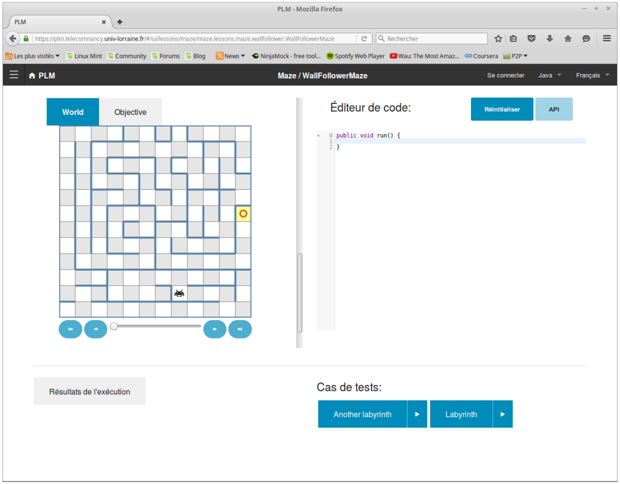
Pick a lesson on the left, and start coding! If unsure, take the "welcome" one: it will guide you through the basics.

Your progress

A graph representing your progress will be displayed here if you select a lesson.

Presentation of PLM

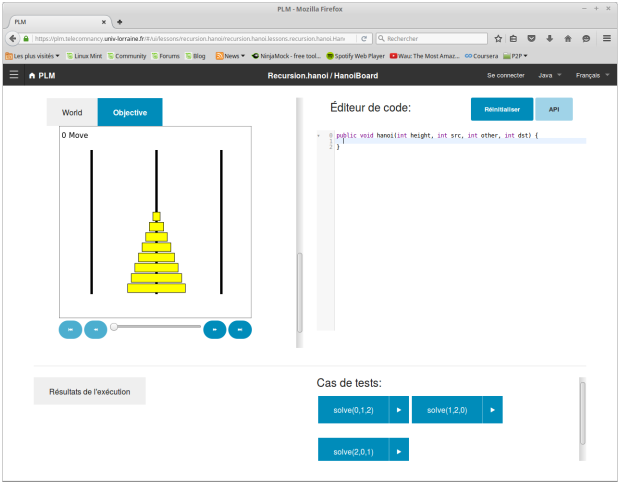
12 lessons, 200 exercises



```
public void run() {
}
```

Presentation of PLM

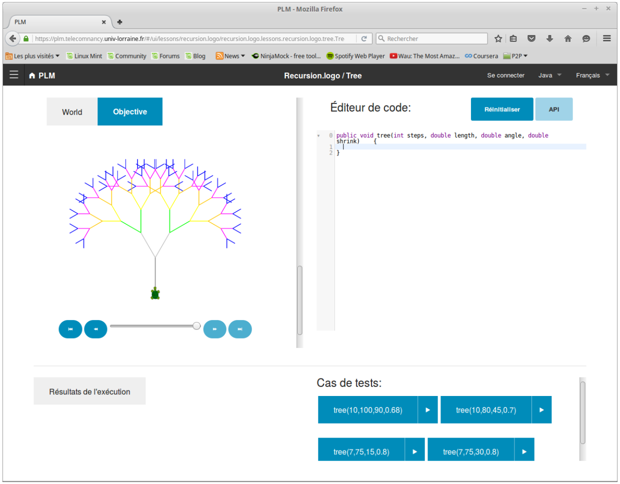
12 lessons, 200 exercises



```java
public void hanoi(int height, int src, int other, int dst) {

}
```

Presentation of PLM

12 lessons, 200 exercises



```
public void tree(int steps, double length, double angle, double
shrink)    {

}
```

Presentation of PLM
Languages and programming languages

- Available languages:
  - English
  - French
  - Brazilian Portuguese
- Supported programming languages:

Java Scala python™

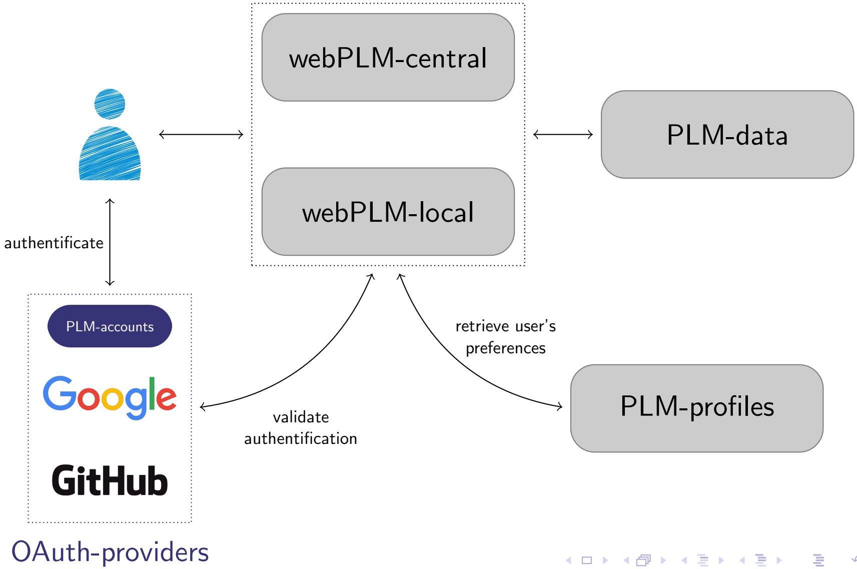
- Formerly a fat client
    - Written in Java

Presentation of PLM
Evolution of the project

- Formerly a fat client
    - Written in Java
- Switch to a web application
    - Headless version of PLM
    - Server implemented in Scala using *PlayFramework*
    - User interface written in Javascript using *AngularJS* and *Foundation*

Presentation of PLM

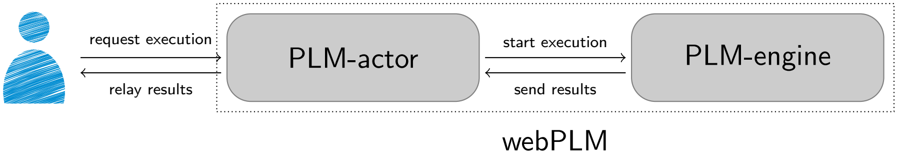
Application's architecture



Outline

Assessment of user's code

Execution components



Assessment of user's code

Limits

- Run on the same machine, same JVM

Assessment of user's code

Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops

Assessment of user's code

Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Endless file creation

Assessment of user's code
Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Endless file creation
- And from *System.exit(whatever)*?

Assessment of user's code

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Endless file creation
- And from *System.exit(whatever)*?
- Scalability issues

Assessment of user's code

Chosen solution

- Delegate the execution to workers
  - Called *Judges* in the litterature
  - Use headless version of PLM as well
  - Execute user's code and send back result to webPLM

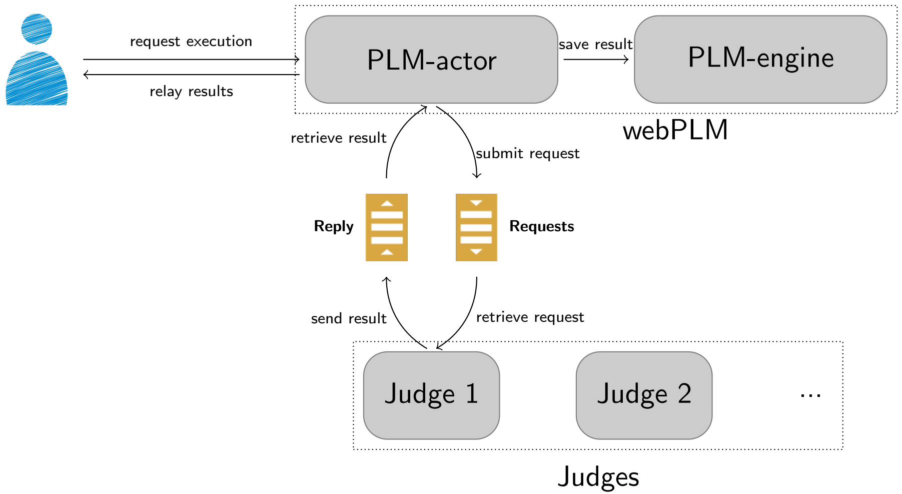
Assessment of user's code

Chosen solution

- Delegate the execution to workers
  - Called *Judges* in the litterature
  - Use headless version of PLM as well
  - Execute user's code and send back result to webPLM

Assessment of user's code

Architecture with judges



Assessment of user's code

Implementation

- Distribute workload using message queues
    - One queue for requests
    - One queue per result

Assessment of user's code

Implementation

- Distribute workload using message queues
    - One queue for requests
    - One queue per result
- *Let it crash* strategy
    - Prevent obvious issues with a security manager
    - Handle timeout and crash

Assessment of user's code

Pros and cons

- Pros:
  - Allow to run code without impacting webPLM's performances
  - Meet the scalability requirements

Assessment of user's code

Pros and cons

- Pros:
  - Allow to run code without impacting webPLM's performances
  - Meet the scalability requirements
- Cons:
  - Make sure to use the right version of PLM
  - Need to deploy them easily
  - Should restart them after each execution
  - Have to restrict their resources usage

Assessment of user's code

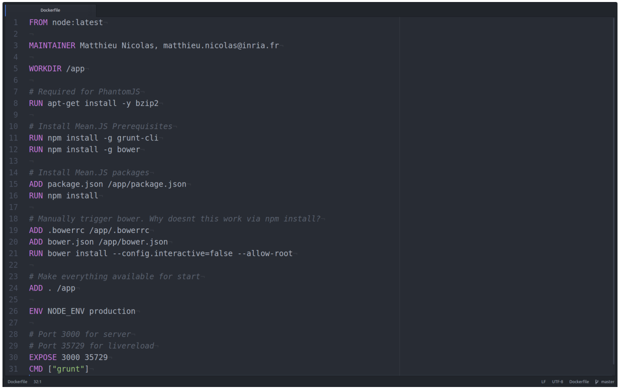
Docker

- Lightweight virtualization tool
- Build image of your application
- Run containers based on images

Assessment of user's code

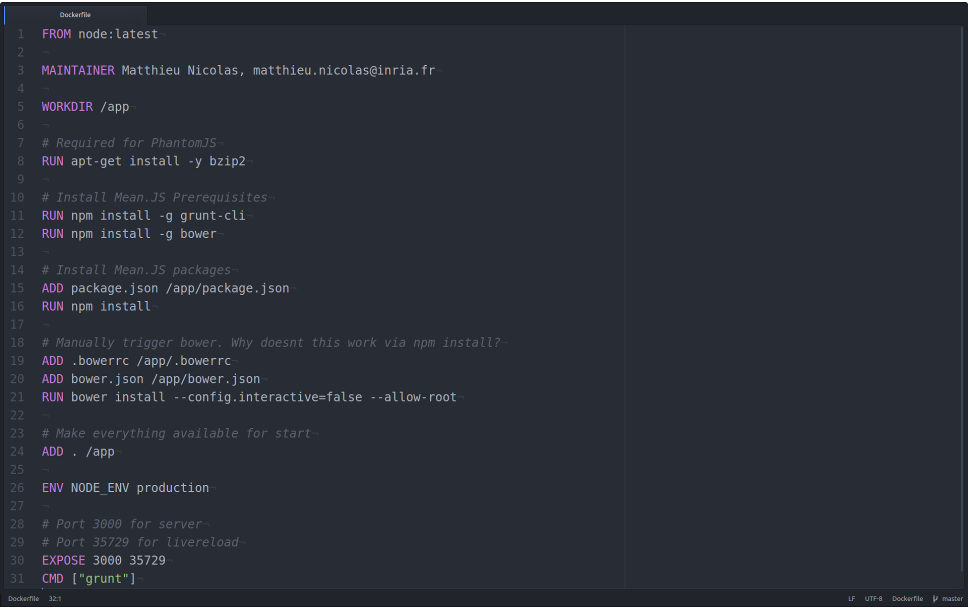
Example of Dockerfile

- Dockerfiles describe how to set up the application



```
FROM node:latest

MAINTAINER Matthieu Nicolas, matthieu.nicolas@inria.fr

WORKDIR /app

# Required for PhantomJS
RUN apt-get install -y bzip2

# Install Mean.JS Prerequisites
RUN npm install -g grunt-cli
RUN npm install -g bower

# Install Mean.JS packages
ADD package.json /app/package.json
RUN npm install

# Manually trigger bower. Why doesnt this work via npm install?
ADD .bowerrc /app/.bowerrc
ADD bower.json /app/bower.json
RUN bower install --config.interactive=false --allow-root

# Make everything available for start
ADD . /app

ENV NODE_ENV production

# Port 3000 for server
# Port 35729 for livereload
EXPOSE 3000 35729
CMD ["grunt"]
```

- Run *docker build -t tag /path/to/Dockerfile* to build the image
- Start containers with *docker run tag*

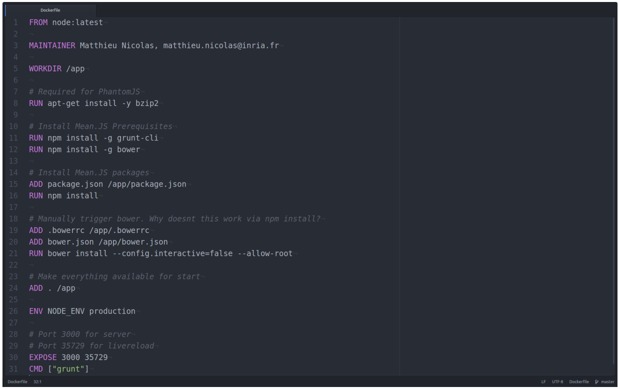

Dockerfile

```dockerfile
FROM node:latest

MAINTAINER Matthieu Nicolas, matthieu.nicolas@inria.fr

WORKDIR /app

# Required for PhantomJS
RUN apt-get install -y bzip2

# Install Mean.JS Prerequisites
RUN npm install -g grunt-cli
RUN npm install -g bower

# Install Mean.JS packages
ADD package.json /app/package.json
RUN npm install

# Manually trigger bower. Why doesnt this work via npm install?
ADD .bowerrc /app/.bowerrc
ADD bower.json /app/bower.json
RUN bower install --config.interactive=false --allow-root

# Make everything available for start
ADD . /app

ENV NODE_ENV production

# Port 3000 for server
# Port 35729 for livereload
EXPOSE 3000 35729
CMD ["grunt"]
```

Assessment of user's code

Example of Dockerfile

- Dockerfiles describe how to set up the application



```
FROM node:latest

MAINTAINER Matthieu Nicolas, matthieu.nicolas@inria.fr

WORKDIR /app

# Required for PhantomJS
RUN apt-get install -y bzip2

# Install Mean.JS Prerequisites
RUN npm install -g grunt-cli
RUN npm install -g bower

# Install Mean.JS packages
ADD package.json /app/package.json
RUN npm install

# Manually trigger bower. Why doesnt this work via npm install?
ADD .bowerrc /app/.bowerrc
ADD bower.json /app/bower.json
RUN bower install --config.interactive=false --allow-root

# Make everything available for start
ADD . /app

ENV NODE_ENV production

# Port 3000 for server
# Port 35729 for livereload
EXPOSE 3000 35729
CMD ["grunt"]
```

- Run *docker build -t tag /path/to/Dockerfile* to build the image
- Start containers with *docker run tag*

Assessment of user's code

More about *docker run*

- Can also manage
  - Ports

Assessment of user's code

More about *docker run*

- Can also manage
  - Ports
  - Volumes

Assessment of user's code
More about *docker run*

- Can also manage
  - Ports
  - Volumes
  - Links between containers

Assessment of user's code

More about *docker run*

- Can also manage
    - Ports
    - Volumes
    - Links between containers
    - Environment variables
    - Runtime constraints on resources
    - Restart policies
    - And a **lot more**

Assessment of user's code

More about *docker run*

- Can also manage
  - Ports
  - Volumes
  - Links between containers
  - Environment variables
  - Runtime constraints on resources
  - Restart policies
  - And a **lot more**

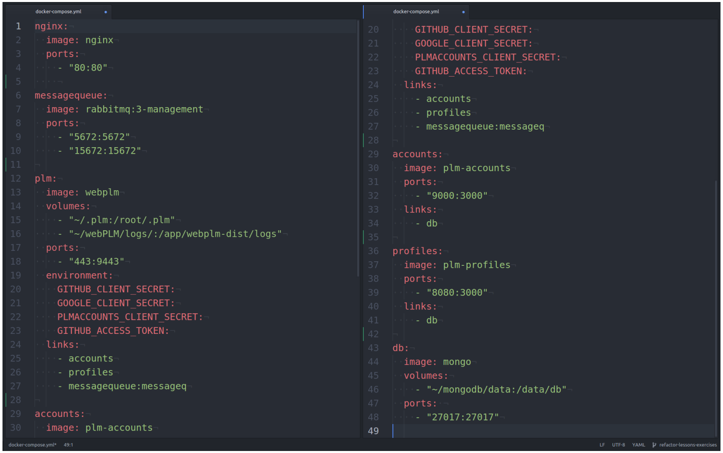
- Commands can become quite complex

```
docker run -p 443:9443 -link plm-accounts:accounts -v
~/webPLM/logs/:/app/webplm-dist/logs webPLM
```

Assessment of user's code

Docker-compose

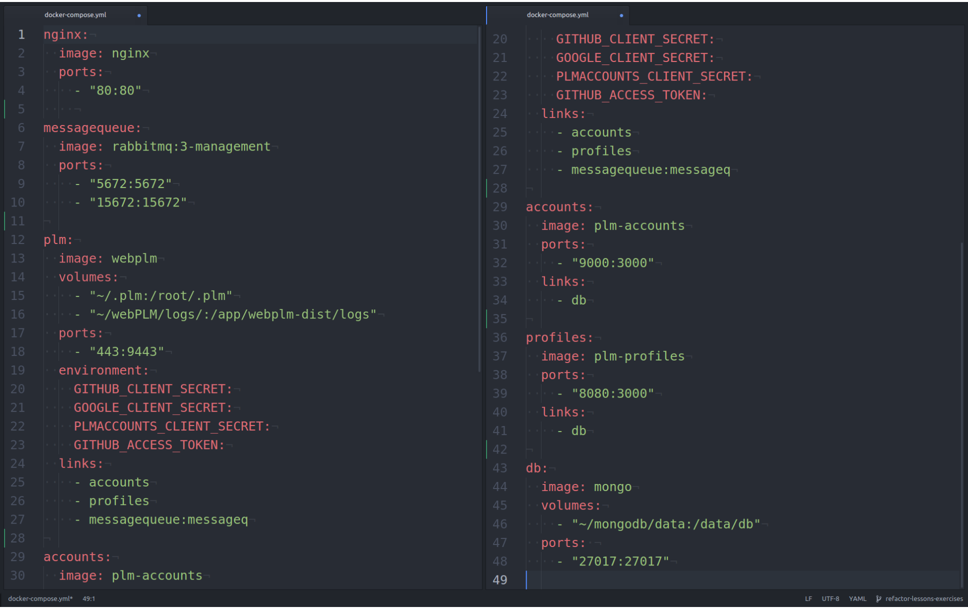
- Tool to easily deploy multi-containers applications



```yaml
nginx:
  image: nginx
  ports:
    - "80:80"
    
messagequeue:
  image: rabbitmq:3-management
  ports:
    - "5672:5672"
    - "15672:15672"

plm:
  image: webplm
  volumes:
    - "~/.plm:/root/.plm"
    - "~/webPLM/logs/:/app/webplm-dist/logs"
  ports:
    - "443:9443"
  environment:
    GITHUB_CLIENT_SECRET:
    GOOGLE_CLIENT_SECRET:
    PLMACCOUNTS_CLIENT_SECRET:
    GITHUB_ACCESS_TOKEN:
  links:
    - accounts
    - profiles
    - messagequeue:messageq

accounts:
  image: plm-accounts
```

```yaml
    GITHUB_CLIENT_SECRET:
    GOOGLE_CLIENT_SECRET:
    PLMACCOUNTS_CLIENT_SECRET:
    GITHUB_ACCESS_TOKEN:
  links:
    - accounts
    - profiles
    - messagequeue:messageq

accounts:
  image: plm-accounts
  ports:
    - "9000:3000"
  links:
    - db

profiles:
  image: plm-profiles
  ports:
    - "8080:3000"
  links:
    - db

db:
  image: mongo
  volumes:
    - "~/mongodb/data:/data/db"
  ports: 
    - "27017:27017"

```

- Deploy environment with *docker-compose up*



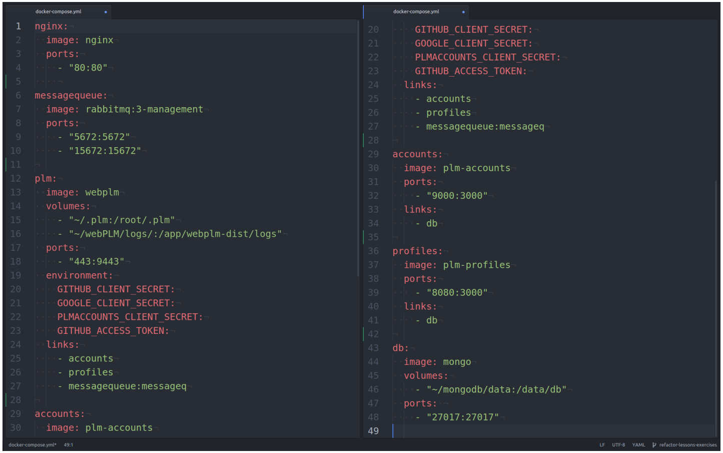
```yaml
nginx:
  image: nginx
  ports:
    - "80:80"

messagequeue:
  image: rabbitmq:3-management
  ports:
    - "5672:5672"
    - "15672:15672"

plm:
  image: webplm
  volumes:
    - "~/.plm:/root/.plm"
    - "~/webPLM/logs/:/app/webplm-dist/logs"
  ports:
    - "443:9443"
  environment:
    GITHUB_CLIENT_SECRET:
    GOOGLE_CLIENT_SECRET:
    PLMACCOUNTS_CLIENT_SECRET:
    GITHUB_ACCESS_TOKEN:
  links:
    - accounts
    - profiles
    - messagequeue:messageq

accounts:
  image: plm-accounts
```

```yaml
    GITHUB_CLIENT_SECRET:
    GOOGLE_CLIENT_SECRET:
    PLMACCOUNTS_CLIENT_SECRET:
    GITHUB_ACCESS_TOKEN:
  links:
    - accounts
    - profiles
    - messagequeue:messageq

accounts:
  image: plm-accounts
```

```yaml
  ports:
    - "9000:3000"
  links:
    - db

profiles:
  image: plm-profiles
  ports:
    - "8080:3000"
  links:
    - db

db:
  image: mongo
  volumes:
    - "~/mongodb/data:/data/db"
  ports:
    - "27017:27017"
```

Assessment of user's code
Docker-compose

- Tool to easily deploy multi-containers applications



```yaml
nginx:
  image: nginx
  ports:
    - "80:80"
    
messagequeue:
  image: rabbitmq:3-management
  ports:
    - "5672:5672"
    - "15672:15672"

plm:
  image: webplm
  volumes:
    - "~/.plm:/root/.plm"
    - "~/webPLM/logs/:/app/webplm-dist/logs"
  ports:
    - "443:9443"
  environment:
    GITHUB_CLIENT_SECRET:
    GOOGLE_CLIENT_SECRET:
    PLMACCOUNTS_CLIENT_SECRET:
    GITHUB_ACCESS_TOKEN:
  links:
    - accounts
    - profiles
    - messagequeue:messageq

accounts:
  image: plm-accounts
```

```yaml
    GITHUB_CLIENT_SECRET:
    GOOGLE_CLIENT_SECRET:
    PLMACCOUNTS_CLIENT_SECRET:
    GITHUB_ACCESS_TOKEN:
  links:
    - accounts
    - profiles
    - messagequeue:messageq

accounts:
  image: plm-accounts
  ports:
    - "9000:3000"
  links:
    - db

profiles:
  image: plm-profiles
  ports:
    - "8080:3000"
  links:
    - db

db:
  image: mongo
  volumes:
    - "~/mongodb/data:/data/db"
  ports:
    - "27017:27017"

```

- Deploy environment with *docker-compose up*

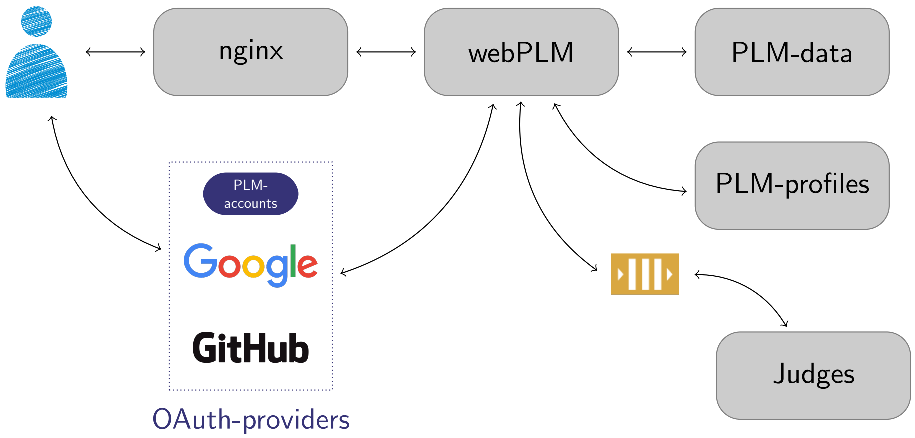
Assessment of user's code

Docker in our case

- Deploy easily all components
- Restart judges automatically
- Limit users' mischiefs

Outline

Result
Current architecture



Result

Live-session in TELECOM Nancy

- Used in TELECOM Nancy in September 2015
- 30 hours of live testing with 100 students.

Result
Live-session in TELECOM Nancy

- Used in TELECOM Nancy in September 2015
- 30 hours of live testing with 100 students.

- Engine is (almost) working fine...
- ... but user experience needs to be improved!

Result

Live-session in TELECOM Nancy

- Can't cope with the workload.

Result

Live-session in TELECOM Nancy

- Can’t cope with the workload.
- No tools for monitoring set up...

Result

Live-session in TELECOM Nancy

- Can't cope with the workload.
- No tools for monitoring set up...
- ... so the bottleneck is unknown.

Outline

Next steps

Refactor the code

- Rushed to release a stable version before September 2015...
- Needed to clean some parts of the code.
- Standardized behavior of local and server mode.

Next steps

Simplify workflow to adapt the content

- Store most of content inside PLM.
- Heavy and error prone workflow.
- Need to extract the content from PLM's jar.
- Allow to implement an exercise editor.

Next steps
Solve performance issues

- Set up some monitoring tools.
- Perform some load testing to identify the bottleneck.

Questions

Thanks for your attention, any questions?